EP 35737 A1

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 November 2001 (22.11.2001)

PCT

(10) International Publication Number
WO 01/88704 A1

(51) International Patent Classification[7]: G06F 9/45, 9/44, 17/28

(21) International Application Number: PCT/SE01/01079

(22) International Filing Date: 16 May 2001 (16.05.2001)

(25) Filing Language: Swedish

(26) Publication Language: English

(30) Priority Data:
0001798-8 16 May 2000 (16.05.2000) SE

(71) Applicant *(for all designated States except US)*: **EUROCITY ONLINE** [SE/SE]; Västra Trädgårdsgatan 17, S-111 53 Stockholm (SE).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **ENGSTRÖM, Hans** [SE/SE]; Kragenäsvägen 13 A, S-181 65 Lidingö (SE).

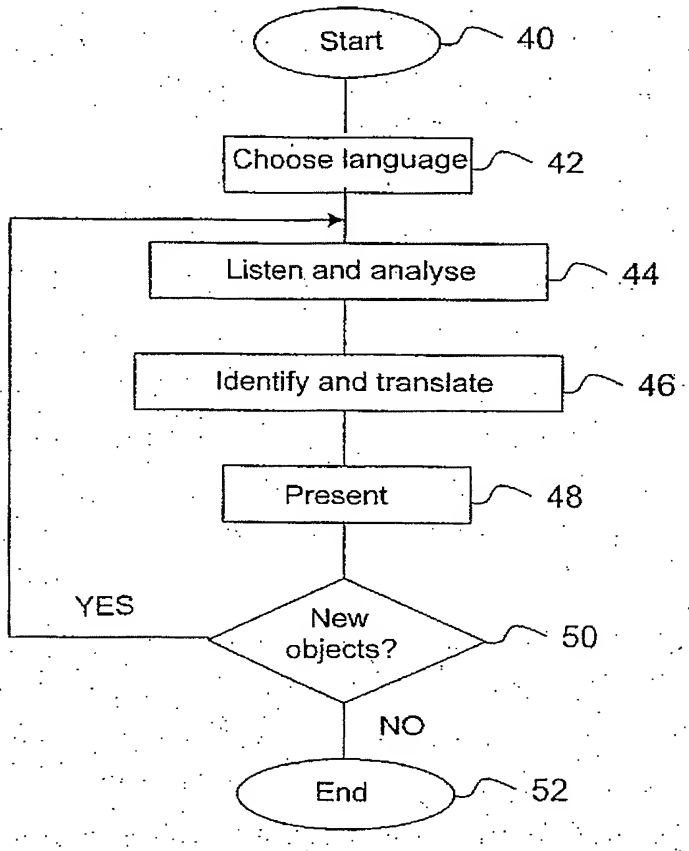
(74) Agents: KARLSSON, Leif et al.; L.A. Groth & Co. KB, Box 6107, S-102 32 Stockholm (SE).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AT (utility model), AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, CZ (utility model), DE, DE (utility model), DK, DK (utility model), DM, DZ, EE, EE (utility model), ES, FI, FI (utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: REAL TIME TRANSLATION OF USER INTERFACES IN AN ARBITRARY APPLICATION



(57) Abstract: The present invention relates to a method for real time translation of user interfaces in any chosen application in a first language, wherein information is transferred from the application to the user interface via a message queue or line. The method uses a first program module which is connected to the message queue by means of hook technology and/or listens to function calls by means of interception technology, a second program module for communication, said second program module being connected to the first program module via a memory device, and software connected to the second program module. The method includes the following steps in which there is chosen for the application a language which is different to the first language; the first program module listens to and analyses the messages in the message queue and/or function calls that include function items; when a unit or an object in a graphic user interface has been identified, this unit or object is translated by the first program module into said language by means of a dictionary stored in the memory device; the translated unit/object is presented in the graphic user interface; and all steps are repeated with the exception of the first step, provided that new units/objects are found, wherein the software controls the execution of these steps.

WO 01/88704 A1

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

REAL TIME TRANSLATION OF USER INTERFACES IN AN ARBITRARY APPLICATION

Field of invention

According to a first aspect, the present invention relates to a method for real time translation of user interfaces in any chosen application.

According to a second aspect, the present invention relates to a computer software for real time translation of a user interface in any chosen application.

According to a third aspect, the present invention relates to at least one computer program element for real time translation of a user interface in any chosen application.

Description of the background art

Localisation of a software product is traditionally effected by translating directly in the source code and compiling a new translated version of the program. This requires access to the source code, meaning that it is carried out many times by the program developers directly.

This means that in order to provide, or obtain, a plurality of different language versions, it is necessary to have access to personnel that have a command of programming to a certain extent and also access to all different languages that shall be localised.

On the other hand, if this work is carried out externally there is a danger that the source code will be lost. Many refuse to do this and release their programs solely in the English language, for instance.

If software that is found in different languages includes a patch or a bugfix, the software will probably not function in more than one original language, or the updated parts of the software will return to the original language.

It is then always necessary to produce a bugfix for respective language version of the program, in order to be able to guarantee all of the same types of corrections.

When a new version of the program is issued, it is also necessary to begin from the beginning and, most often, to carry out completely new localisations, since it is seldom that a very old particular program code can be used.

Since the issue of new programs is becoming more and more frequent, a large number of resources is required for holding all language versions alive.

Another way of localising software is to open compiled EXE and DLL files with external software and to make the translation there. This involves roughly the same amount of work as that involved when translating in the source code.

This may be easier in many instances, since it is slightly more visual, although there is a danger that the files will not be fully compatible with one another, as they have not been compiled at one and the same time.

In respect of bugfixes or new versions of the program, the problems met with are the same as those when translating in the source code.

One problem that can occur, depending on which external software is used and which program is translated, is that it is necessary to trust that no undesirable items that may cause crashes have been written into the files.

Summary of the invention

An object of the present invention is to solve the aforementioned problems.

This is achieved in accordance with a first aspect of the present invention with a method for real time translation of user interfaces in any chosen application in a first language according to claim 1. Information is transferred from the application via a message queue or line. The method uses a first program module which is connected to the message queue by means of hook technology and/or which listen to a function call by means of interception technology, a second program module for communication, this second program module being connected to the first program module via a memory device, and software connected to the second program module. The method comprises the steps in which

- there is chosen for the application a language which is different than the first language;
- the first program module listens to and analyses the messages in the message queue and/or function calls including function items;
- when a unit or an object in a graphic user interface has been identified, this object or unit is translated by the first program module to said different language with the aid of a dictionary stored in the memory device;
- the translated unit/object is presented in the graphic user interface; and
- all steps are repeated with the exception of the first step provided that new units/objects are present, wherewith the software initiates the execution of these steps.

A number of other advantages are gained, in addition to eliminating the aforesaid problems. Firstly, there is no danger of a code being obtained by unauthorised persons, or of errors being caused in the original program in any other way.

The problems associated with new versions or bugfixes are also overcome, since the inventive method is based on an external dictionary, and all that need be supplemented are the new items in the new version of the software. The remainder is reused.

Because nothing in the source code is changed, it is possible to issue localised versions of the program in several different languages more quickly.

Another important advantage is that translations made on other programs can be reused, therewith saving a significant amount of time and obtaining unitary terminology which enables the user to acquaint himself/herself more easily between different programs and versions.

Another advantage in this context is that if a number of different-language dictionaries are stored in the memory device, each dictionary can be used for translating between said first language and another language.

In this regard, an advantage is afforded when the software carries out the steps in which:

- the software is started when there is chosen for the application a language that is different from the first language;
- if there is an existing application, the application is started automatically or manually;
- when the answer to the preceding step is *Yes*, the second program module for said communication is loaded;
- an initiation is carried out with the name of the dictionary concerned, which is determined by the language chosen;
- if the application is active, there is coupled a hook which starts the first program module indirectly;
- the hook-handle concerned is transferred to the first program module; and
- the software stops in response to a stop message from the first program module.

In this respect, there is gained an advantage when the second program module carries out the steps in which:

- the second program module is started by the software;

- the relevant dictionary is stored in response an initiating message received from the software, said dictionary being determined by the chosen language and the current hook-handle in the memory device; and
- stopping the second program module when the thread in question, in other words the path followed by a process, is stopped.

In this regard, an advantage is gained when the first program module carries out the following steps in which:

- the first program module is started by the software;
- the relevant dictionary is loaded from the memory device and sorted, the relevancy of said dictionary being determined by the language chosen;
- a unit or an object in a graphic user interface is identified when listening to a message in the message queue and/or listening to a functional item;
- said unit or object is translated into the language chosen;
- the nearest two preceding steps are repeated until all of the units or the objects in the graphic user interface have been translated; and
- a stop message is sent to the software.

An advantage is afforded in this connection, when the units or objects can be comprised of window titles, menus or dialogue boxes.

An advantage is also afforded in this connection when the first program module also carries out the step of adapting the size of the translated unit/objects so that they can be accommodated in the graphic user interface.

One advantage in this respect is gained when respective first and second program modules are called as a dynamic link by means of DLL technology.

Another object of the present invention is to provide at least one computer software that can be loaded directly into the internal memory of at least one digital computer. The at least one computer software includes software code portions for carrying out the steps according to the inventive method when said at least one product is run on said at least one computer. A number of further advantages are gained, in addition to eliminating the aforesaid problems. There is no danger of a code emerging into the hands of unauthorised persons, or of causing errors in the original program in some other way.

Those problems that occur with new versions or bugfixes are avoided, since the computer software according to the invention is based on an external dictionary, word list, glossary or like vocabulary, and all that needs to be supplemented are the new features in the new version. The remainder is reused.

By not changing anything in the source code, localised versions of the program can be published in several languages more quickly.

Another important advantage is that it is also possible to reuse translations made on other programs and therewith save a great deal of time and obtain unitary terminology, which enables the users to achieve more readily a self-awareness between different programs and versions.

Another object of the present invention is to provide at least one computer program element that includes computer program code elements for causing at least one computer to execute procedures in which:

- the computer program element or elements is/are started in response to choosing for an application a language which is different from the first language;
- starting the application automatically or manually, if an application exists;
- when the answer to the preceding step is *Yes*, loading a second program module for communication;
- initiating with the name of the dictionary concerned, which is determined by the chosen language;
- if the application is active, couple a hook which indirectly starts a first program module;
- transfer the hook-handle concerned to the first program module; and
- stopping the computer program element or elements in response to a stop message received from the first program module.

A further object of the present invention is to provide a computer program element that includes computer program code elements for causing a computer to execute the following procedures:

- start the computer program element by a software product;
- storing the name of the dictionary concerned, determined by the chosen language, in response to receiving an initiation message from the software, and also the hook-handle in question; and
- stopping the computer program element when a relevant thread is stopped, i.e. the path followed by the process.

Another object of the present invention is to provide at least one computer program element that includes computer program code means for causing at least one computer to execute the following procedures:

- starting the computer program element by a piece of software;
- loading the dictionary concerned, determined by the language chosen, from a

memory device, and sorting said dictionary;

- identifying a unit or an object in a graphic user interface, when a message has been listened to in the message queue between a user interface and an application and/or when a functional item has been listened to;
- translating said unit or object into the chosen language;
- repeating the two immediately preceding steps until all units or objects have been translated; and
- sending a stop message to the software.

It is emphasised that the terms "includes/including" as used in the present description are meant to indicate the presence of given features, steps or components, although it is not meant to exclude the presence of one or more other features, units, steps, components or groups thereof.

The invention will now be described with reference to exemplifying embodiments thereof and also with reference to the accompanying drawings.

Brief description of the drawings

Figure 1 illustrates schematically the interaction between an application and the interfaces seen by the user, in accordance with the present state of the art.

Figure 2 illustrates schematically the interaction between an application and the interfaces seen by the user, in accordance with one embodiment of the present invention.

Figure 3 is a flowchart illustrating a method for real time translation of user interfaces in any chosen application in a first language in accordance with the present invention.

Figure 4 is a flowchart illustrating the steps executed by software in accordance with the present invention.

Figure 5 is a flowchart illustrating the steps carried out by a second program module in accordance with the invention.

Figure 6 is a flowchart of the steps carried out by a first program module in accordance with the present invention.

Figure 7 is a schematic illustration of computer program products and computer program elements according to the present invention.

Detailed description of embodiments

Figure 1 illustrates schematically the interaction between an application and the user interface that the user sees, in accordance with the present state of the art. Fig. 1 shows a chosen application 10 which is connected to a user interface 12 via a message queue or line A, or which belongs to said user interface. The user interface 12 may, for instance, comprise menus, window text, and so on. The application 10 sends a quantity of information to the user interface 12, via the message queue A, this information including, among other things, texts for different elements on a computer screen. The messages are coded with a type designation. The remaining message content varies in accordance with type.

Figure 2 illustrates schematically the interaction between an application and the interfaces seen by the user, in accordance with one embodiment of the present invention. Fig. 2 illustrates a chosen application 20 and a user interface 22, such as in Fig. 1. The chosen application 20 may, for instance, be a chosen PC computer program of the Windows-type. On the other hand, the message queue has a different configuration. In this case, the message queue is divided into a first part A_1 and a second part A_2 of the message queue. In addition, Fig. 2 shows a first program module 24 which hooks into the message stream in an existing program, with the aid of so-called hook technology. When securely in place, the hooked-in program module can spy on the messages and possibly modify the same, and then forward said messages to the user interface, for instance. In this case, the first part A_1 of the message queue passes between the application 20 and the first program module 24, and the second part A_2 of the message queue passes from the first program module 24 and the user interface 22. Fig. 2 also shows a second program module 26 for communication. The two program modules 24, 26 are connected via a common memory device 28. Fig. 2 also shows software 30 which is connected to the second program module 26. In this application, the aforesaid modification consists of a translation. The solution illustrated in Fig. 2 enables simultaneous translation of user interfaces in any selected application. The translation can be made at the right moment in time, by analysing the message flow between the application 20 and the user interface 22.

The interaction illustrated in Fig. 2 between an application and the interfaces seen by the user is suitable when interception technology is used to make a simultaneous translation of user interfaces in any selected applications. Interception technology means that a call from a program to the functions incorporated in

the operation system is controlled so that they pass the additional or supplementary function on the way.

Each program includes a table listing the external functions used by the program, i.e. those functions that are used in the operating system or supplementary modules. The table includes the addresses of respective functions and the name of the module to which they belong. Interception means, quite simply, that the table or the program concerned is scanned and the functions of interest looked for (the supplementary modules of the program must also be looked through). When finding an interesting function, the address in the table is replaced with the address of an own developed function, which will then be called instead of the original function. The own function is therewith able to spy on those texts that are found appropriate, and, when necessary, translate the texts before they are released to the original function (the address of which can thus be saved).

When using interception technology, A_1 and A_2 in Fig. 2 do not represent a message queue, although the function is the same.

According to the present invention, there can be used solely hook technology, solely interception technology or a combination of these two technologies.

It is as though the user has an extra addition of the program. When starting the program, the user chooses whether he will start the original program or a translated version. When the user elects to start a translated program, it is the software 30 that is started in actual fact. However, the software 30 starts directly the target program and adds the first program module 24 including the translation. As a result, the user sees only that a translated program has been started. The user then chooses different functions in the application which are translated prior to being presented on the screen.

Figure 3 is a flowchart illustrating a method for real time translation of user interfaces in any chosen application in a first language in accordance with the invention. The method begins at block 40. The method functions in an environment such as that shown in Fig. 2, i.e. in an environment which includes a respective first and second program module, and a software product. The method continues with the step at block 42, in which there is chosen for the application a language that is different from the first language. No translation is required, of course, if so is not the case. The next step is carried out at block 44, in which the first program module listens to and analyses messages in the message queue and/or function calls including function items. The method continues at block 46 with the step in

which when a unit or an object in a graphic user interface has been identified, this unit or object is translated by the first program module into said language with the aid of a dictionary stored in the memory device. It will be noted that this dictionary does not primarily include single words, but also includes phrases and complete sentences. The next step is commenced at block 48 and consists in presenting the translated unit or object in the graphic user interface. At block 50, there is asked the question of whether or not new units/objects that shall be translated occur. If the answer to this question is *Yes*, steps 42-50 are repeated. On the other hand, if the answer is *No*, the method is terminated at block 52. It will also be noted that it is the software that controls the execution of these steps. A number of different dictionaries are stored in the memory device, wherewith each dictionary is used for translating between said first language and a different language.

Figure 4 is a flowchart illustrating the steps executed by software in accordance with the invention. The software used is the software 30 shown in Fig. 2. The method starts at block 60. The method then continues at block 62 with the software being started when choosing for the application a language that differs from the first language. The question is raised at block 64 as to whether or not applications exist. When the answer to this question is *No*, the method is terminated at block 76. If an application exists, the application is started automatically or manually. If, on the other hand, the answer is *Yes*, the method continues with block 66, wherein the second program module for communication is loaded. The next step, commenced at block 68, comprises an initiation process with the name of the dictionary concerned, said dictionary being determined by the language chosen. The method continues at block 70 with the step of coupling to the application, if active, a hook which indirectly starts the first program module. The next step, at block 72, comprises transferring the hook-handle in question to the first program module. A handle is a number or a value that can be used to obtain access to a unit or to an object, such as a file, a window or a dialog box, in a graphic user interface. The method then continues at block 74 with stopping the software when it receives a stop message from the first program module. The method is terminated at block 76.

Figure 5 is a flowchart illustrating the steps executed by the second program module (c.f. Fig. 2) in accordance with the present invention. The method begins at block 80. At block 82, the method continues with the step in which the second program module is started by the software. The next step, at block 84,

comprises storing a relevant dictionary and the current hook-handle in the memory device in response to an initiation message received from the software, said dictionary being determined by the language chosen. The method continues at block 86 with the step in which the second program module stops when a relevant thread is stopped, i.e. when the path followed by a process is stopped. The method is terminated then at block 88.

Figure 6 is a flowchart illustrating the steps executed by the first program module (c.f. Fig. 2) in accordance with the present invention. The method commences at block 90. The method then continues at block 92 with the step in which the first program module is started by the software. The next step, at block 94, comprises loading the relevant dictionary from the memory device and sorting the dictionary, said dictionary being determined by the language chosen. Sorting of a dictionary involves reading phrases from the dictionary and sorting them in an alphabetic order in the RAM memory of the computer, for instance, so as to facilitate searching for phrases in the translation phase. The method then continues at block 96 with the question of whether or not a message in the message queue and/or a functional item has been listened to. If the answer to the question is *No*, the block 96 is re-executed. If, on the other hand, the answer is *Yes*, the method continues with block 98 in which a unit or an object in a graphic user interface is identified. The next step, at block 100, comprises translating said unit or object into the language chosen. The method then continues at block 102 with a question asking whether all units or objects in the graphic user interface have been translated. If the answer to the question is *No*, steps 98-102 are repeated. If, on the other hand, the answer is *Yes*, the method continues with block 104, wherein a stop message is sent to the software. The method then terminates at block 106.

The aforesaid units or objects may consist of window titles, menus or dialog boxes, for instance.

An advantage is also gained when the first program module carries out the step of adapting the size of the translated units/objects so that they can be accommodated in the graphic user interface. This is particularly important when a tool that includes text is present in the user interface, since the space required, e.g., for a word is limited and because a translated word may be much longer than the original word in the original language.

The respective first and second program modules are called by means of DLL technology, which is a dynamic link.

Figure 7 is a schematic illustration of some items of computer program products in accordance with the present invention. The figure shows different digital computers 108_1,..., 108_n, where n is an integer. The figure also shows n-number of different computer program products 110_1,..., 110_n, shown here in different forms of compact disks. The various computer program products 110_1,..., 110_n can be loaded directly into the internal memory of respective digital computers 108_1,..., 108_n. Each computer program product 110_1,..., 110_n includes software code parts for executing some or all steps according to Fig. 3, when the computer program product/products 110_1,..., 110_n are run on the computer/computers 108_1,..., 108_n. The computer program products 110_1,..., 110_n may have the form of diskettes, RAM disks, magnetic tapes, optomagnetic disks or some other suitable products.

Also shown in Fig. 7 are a number of computer program elements 112_1,..., 112_n, where n is an integer. The computer program element 112_1,..., 112_n includes computer program code means for causing at least one computer to execute the steps according to Figs. 4, 5 or 6.

The computer program element/elements may be recorded on at least one computer readable medium.

It will be understood that the invention is not restricted to the aforedescribed embodiments and that many modifications can be made within the scope of the accompanying claims.

CLAIMS

1. A method for real time translation of user interfaces in any chosen application in a first language, wherein information is transferred from the application to the user interface via a message queue or line, said method using a first program module which is connected to a message queue by means of hook technology and/or listens to functions calls through the medium of interception technology, a second program module for communication, wherein the second program module is connected to the first program module via a memory device, and software connected to the second program module, said method comprising the following steps in which

- a language different from the first language is chosen for the application;
- the first program module listens to and analyses the messages in the message queue and/or function calls that include function items;
- when a unit or an object in a graphic user interface has been identified, the unit or object is translated by the first program module into said language with the aid of a dictionary stored in the memory device;
- the translated unit/object is presented in the graphic user interface; and
- in which all steps are repeated with the exception of the first, provided that new units/objects are found, wherewith the software controls the execution of these steps.

2. A method for real time translation of user interfaces in any chosen application in a first language according to claim 1, **characterized** in that a number of different dictionaries are stored in the memory device, wherein each dictionary is used for translation between said first language and a second language.

3. A method for real time translation of user interfaces in any chosen application in a first language according to any one of claims 1-2, **characterized** in that the software executes the following steps:

- the software is started when choosing for the application a language that is different to the first language;
- the application is started automatically or manually if an application exists;
- if the answer to the preceding step is *Yes*, the second communication program module is loaded;

- the relevant dictionary is initiated by name, the relevance of said dictionary being determined by the language chosen;
- a hook is coupled if the application is active, said hook starting the first program module indirectly;
- the hook-handle concerned is transferred to the first program module; and
- the software stops upon receipt of a stop message from the first program module.

4. A method for real time translation of user interfaces in any chosen application in a first language in accordance with any one of claims 1-3, **characterized** in that the second program module executes the following steps in which:

- the second program module is started by the software;
- when an initiation message has been received from the software, the name of the relevant dictionary and the current hook-handle are stored in the memory device, said dictionary being determined by the language chosen; and
- the second program module stops when a relevant thread is stopped, i.e. when the path followed by the process stops.

5. A method for real time translation of the user interfaces in any chosen application in a first language in accordance with any one of claims 1-4, **characterized** in that the first program module executes the following steps in which:

- the first program is started by the software;
- the relevant dictionary is loaded from the memory device and sorted, the relevancy of said dictionary being determined by the language chosen;
- if a message in the message queue has been listened to and/or a function item has been listened to, a unit or an object in a graphic user interface is identified;
- the unit or the object is translated to the language chosen;
- the two nearest preceding steps are repeated until all units or objects in the graphic user interface have been translated; and
- a stop message is sent to the software.

6. A method for real time translation of user interfaces in any chosen application in a first language in accordance with any one of claims 1-5, **characterized** in that the units or objects may be comprised of, for instance, window titles, menus or dialog boxes.

7. A method for real time translation of user interfaces in any chosen application in a first language in accordance with claim 5, **characterized** in that the first module also executes the step

- of adapting the size of the translated units/objects so that they can be accommodated in the graphic user interface.

8. A method for real time translation of user interfaces in any chosen application in a first language in accordance with any one of claims 1-7, **characterized** respective first and second program modules are called by means of DLL technology as a dynamic link.

9. At least one computer software (110_1,..., 110_n) that can be loaded directly into the internal memory of at least one digital computer (108_1,..., 108_n) comprises software code parts for executing the steps according to claim 1, where said at least one product (110_1,..., 110_n) is run on said at least one computer (108_1,..., 108_n).

10. At least one computer program element (112_1,..., 112_n) comprising computer program code means for causing at least one computer to execute the following steps in which

- the computer program element or elements is/are started when there is chosen for an application a language that is different to the first language;
- if an application exists, it is started automatically or manually;
- if the answer to the preceding step is *Yes*, a second communication program module is loaded;
- the name of the relevant dictionary is initiated, the relevancy of said dictionary being determined by the language chosen;
- if the application is active there is coupled a hook which starts a first program module indirectly;
- the hook-handle concerned is transferred to the first program module;
- the computer program element or elements stops when it/they receive a stop message from the first program module.

11. A computer program element (112) comprising computer program code

means for causing a computer to execute the following steps in which

- the computer program element is started by software;
- when an initiation message has been received from the software, the name of the relevant dictionary and the hook-handle concerned are stored in a memory device, the relevancy of said dictionary being determined by the language chosen; and
- the computer program element stops when a relevant thread is stopped, i.e. the path followed by a process.

12. At least one computer program element (112_1,..., 112_n) comprising computer program code means for causing at least one computer to execute the following steps in which

- the computer program element is started by software;
- the relevant dictionary is loaded from a memory device and the dictionary sorted, the relevancy of said dictionary being determined by the language chosen;
- if a message in a message queue between a user interface and an application has been listened to and/or a function item has been listened to, a unit or an object in a graphic user interface is identified;
- said unit or object is translated into said chosen language;
- the two nearest preceding steps are repeated until all units or objects have been translated; and
- a stop message is sent to the software.

13. At least one computer program element (112_1,..., 112_n) according to at least one of claims 10-12 recorded on at least one computer readable medium.

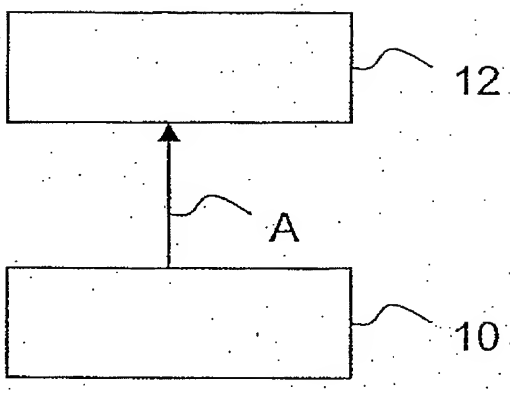


FIG. 1

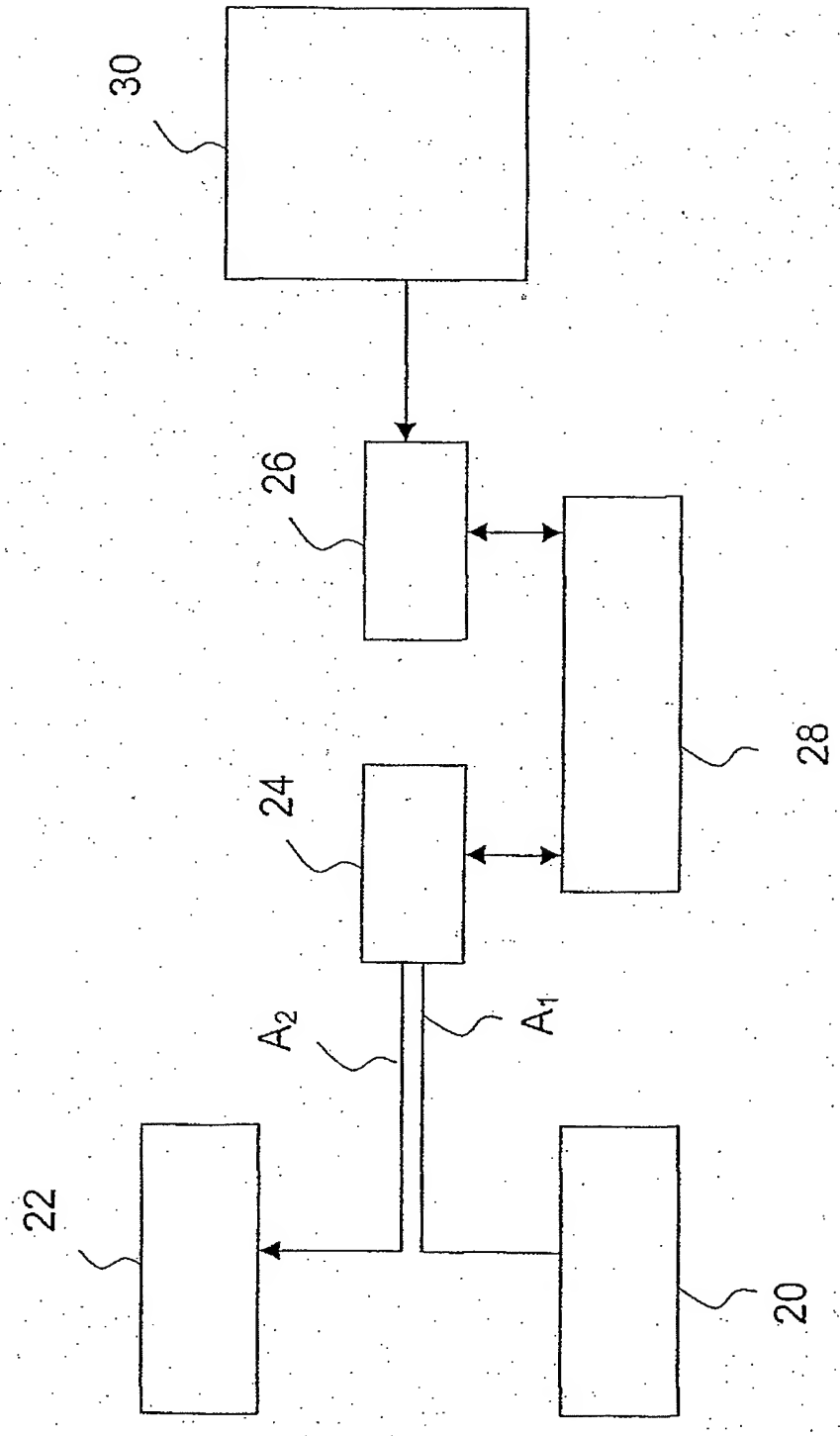


FIG. 2

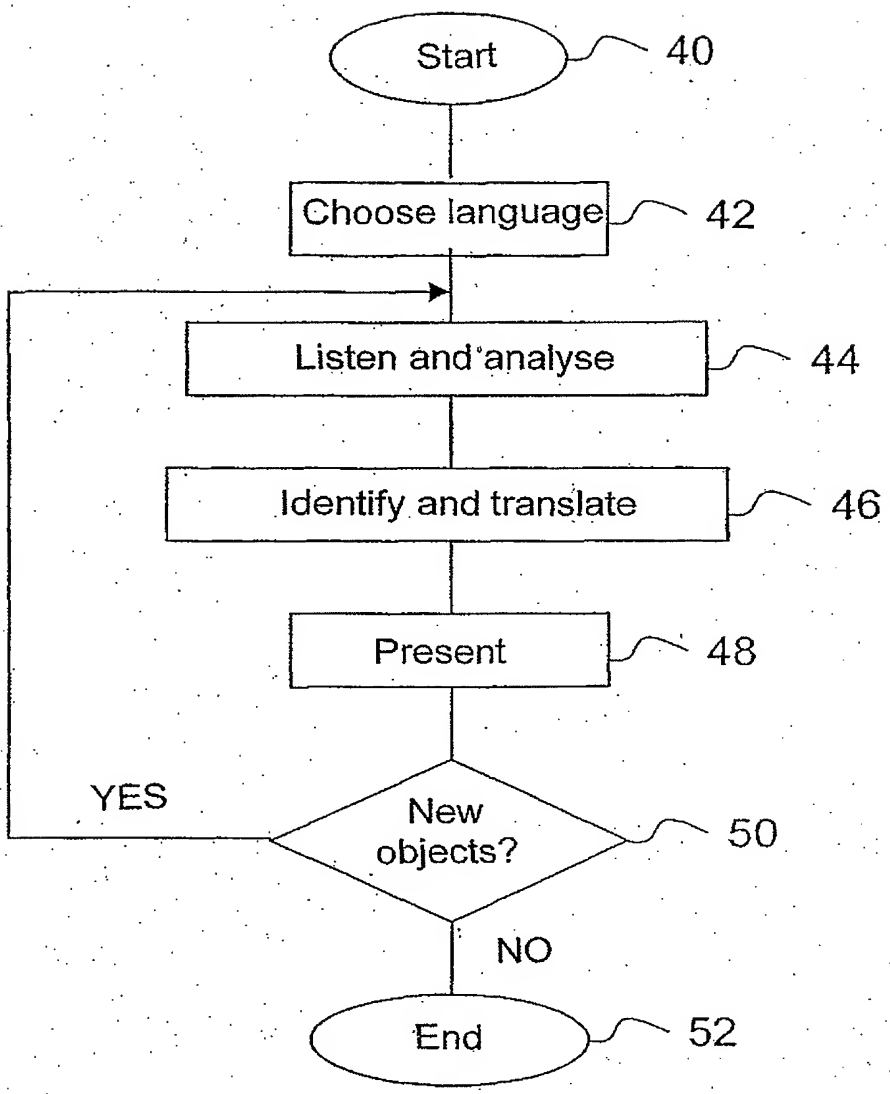

Start
40
Choose language
42
Listen and analyse
44
Identify and translate
46
Present
48
YES
New objects?
50
NO
End
52

FIG. 3

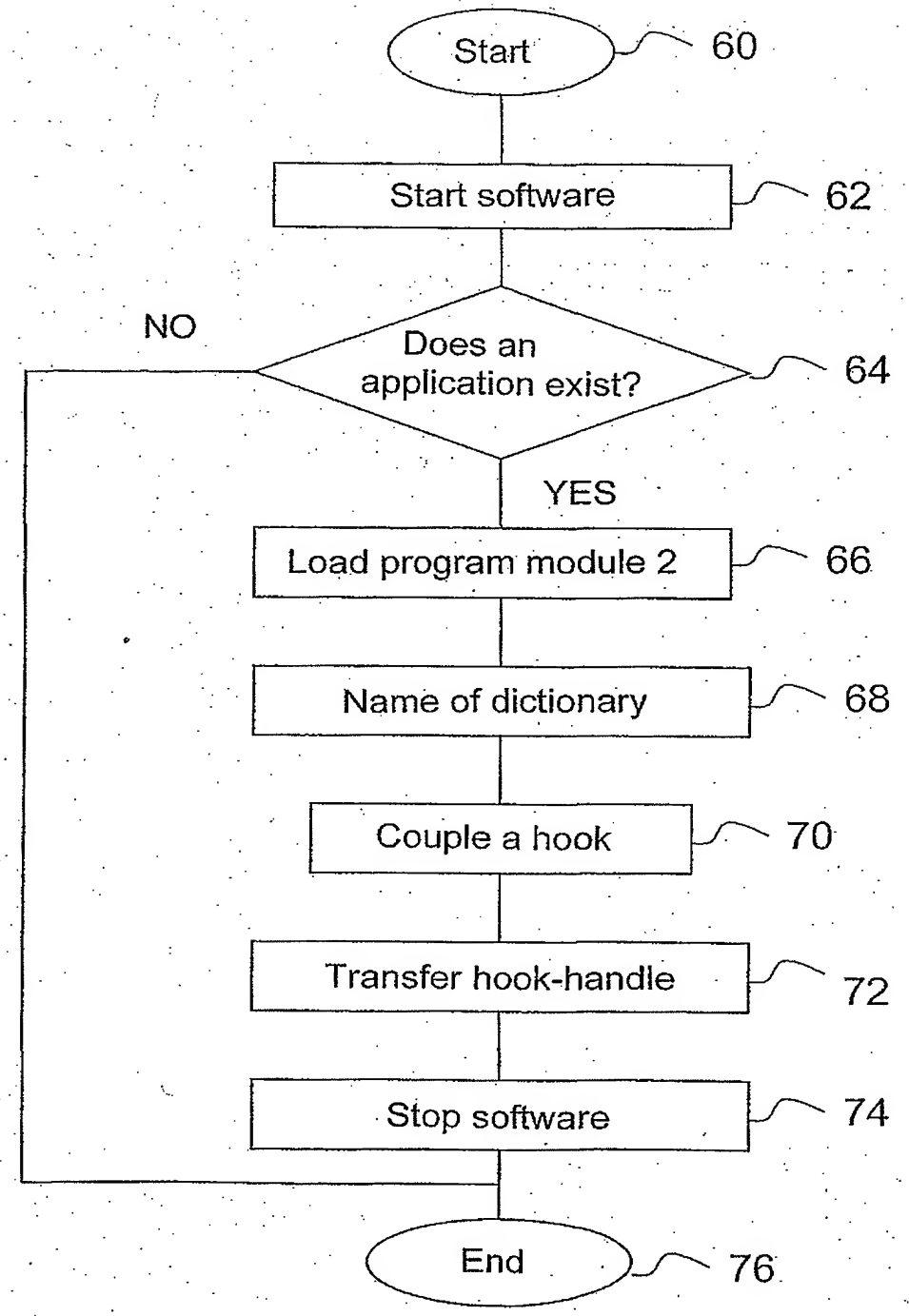

Start
60
Start software
62
NO
Does an application exist?
64
YES
Load program module 2
66
Name of dictionary
68
Couple a hook
70
Transfer hook-handle
72
Stop software
74
End
76

FIG. 4

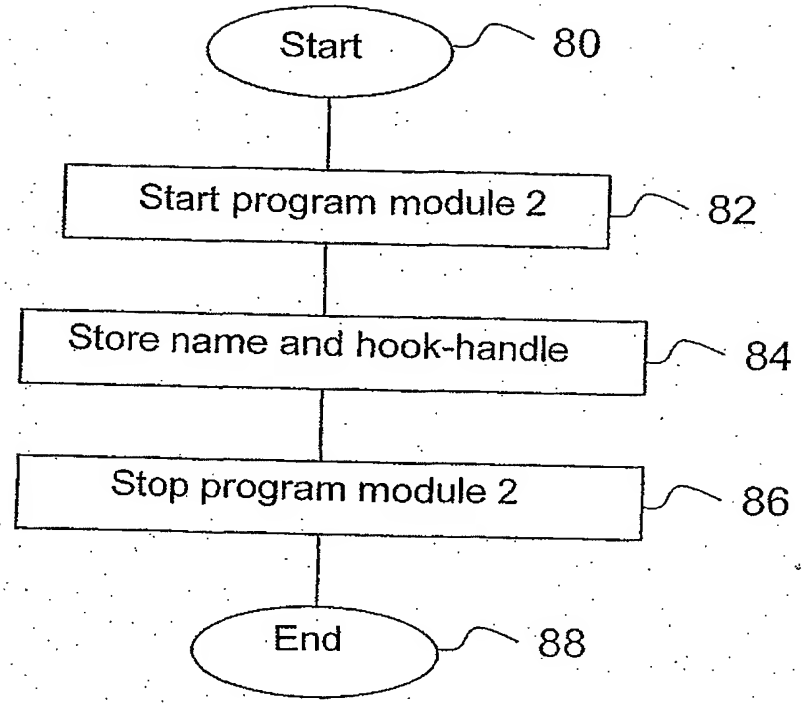

Start
80
Start program module 2
82
Store name and hook-handle
84
Stop program module 2
86
End
88

FIG. 5

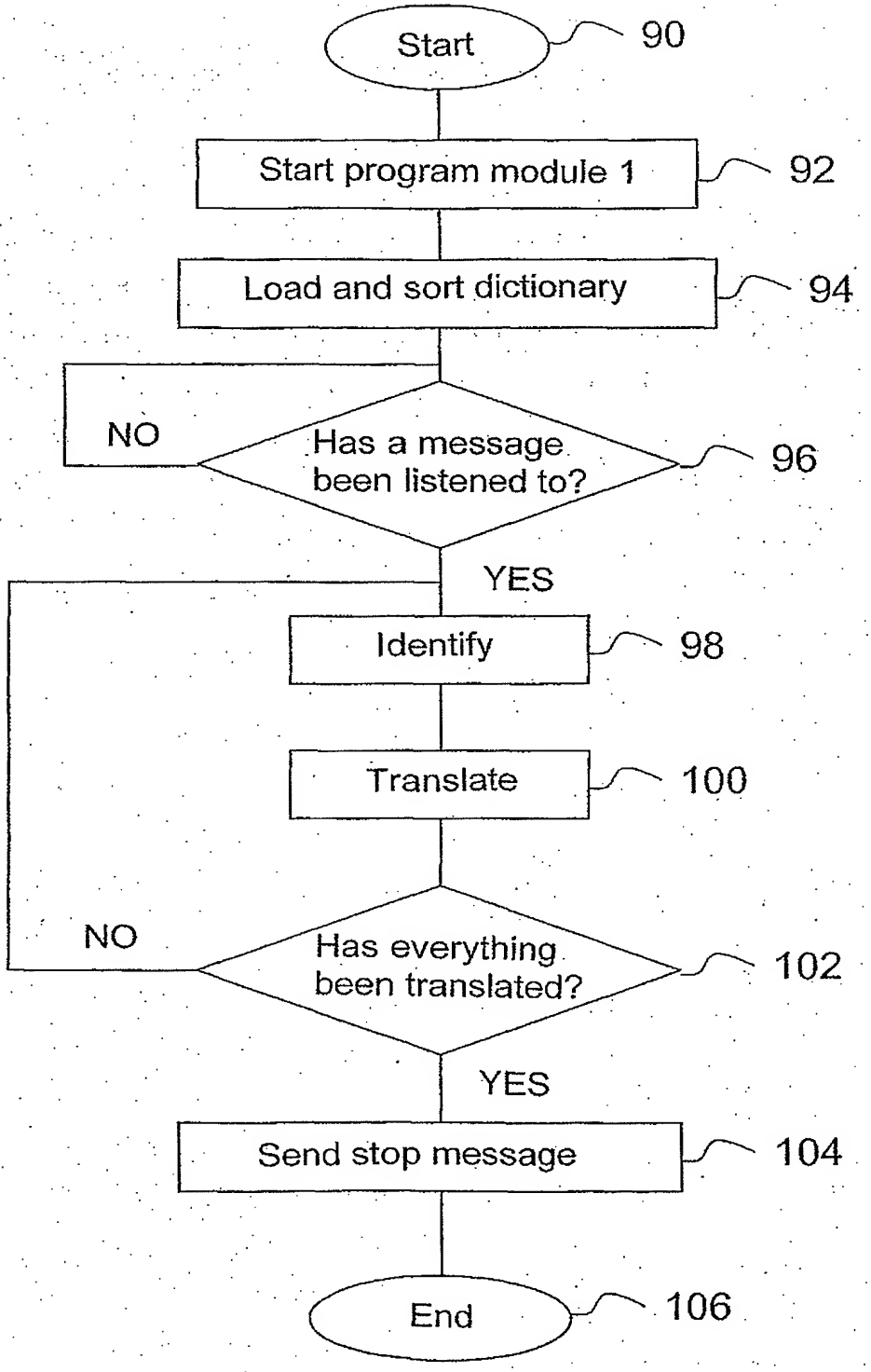

Start
90
Start program module 1
92
Load and sort dictionary
94
NO
Has a message been listened to?
96
YES
Identify
98
Translate
100
NO
Has everything been translated?
102
YES
Send stop message
104
End
106

FIG. 6

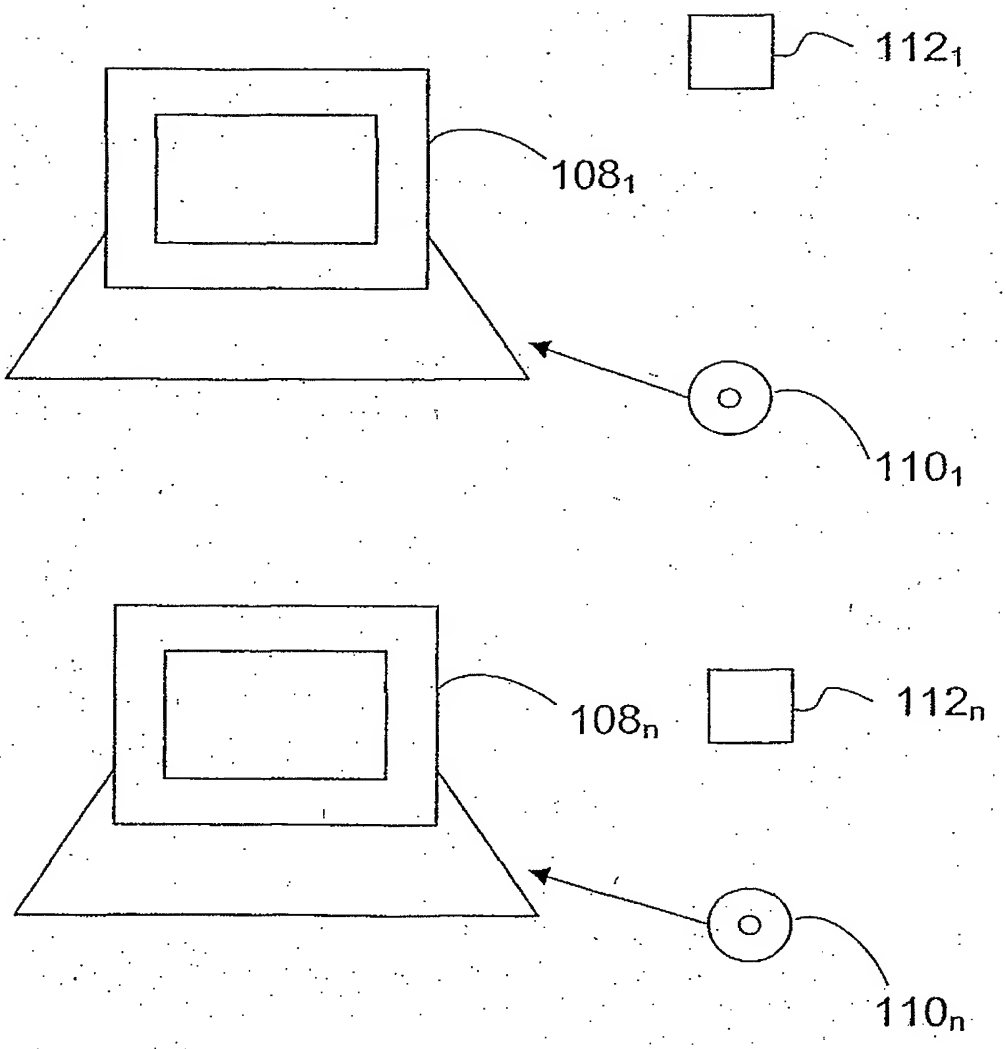


FIG. 7

INTERNATIONAL SEARCH REPORT

In...ional application No.
PCT/SE 01/01079

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: G06F 9/45, G06F 9/44, G06F 17/28
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5675804 A (JUDIANTO SIDIK ET AL), 7 October 1997 (07.10.97), figure 3, claims 2, 10-14 --	1-13
X	US 5579520 A (JOHN G. BENNETT), 26 November 1996 (26.11.96), column 1, line 18 - line 66; column 2, line 1 - line 11; column 9, line 35 - line 50 --	1-13
P,X	US 6086622 A (YAYOI ABE ET AL), 11 July 2000 (11.07.00), column 8 --	1-13
A	US 5748961 A (CHRISTINE BETH HANNA ET AL), 5 May 1998 (05.05.98), column 1, figure II --	1-13

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 August 2001	24 -08- 2001
Name and mailing address of the ISA: Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Fernando Farieta/EÖ Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 01/01079

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0929034 A2 (NEC CORPORATION), 14 July 1999 (14.07.99), figure 3A --	1-13
A	US 5966539 A (AMITABH SRIVASTAVA), 12 October 1999 (12.10.99), column 1; column 2; column 14, figure 2 --	1-13
A	WPI/Derwent's abstract, Accession no. 1996256638, week 9626, ABSTRACT OF JP, 8106444 (NIPPON DENKI ENG KK) 23 April 1996 (23.04.1996) --	1-13
A	Patent Abstracts of Japan, abstract of JP 62-177628 A (TOSHIBA CORP), 4 August 1987 (04.08.87) --	1-13
A	Patent Abstracts of Japan, abstract of JP 62-171036 A (FUJITSU LTD), 28 July 1987 (28.07.87) --	1-13

INTERNATIONAL SEARCH REPORT

Information on patent family members

02/08/01

Int...ional application No.
PCT/SE 01/01079

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5675804 A	07/10/97	NONE	
US 5579520 A	26/11/96	NONE	
US 6086622 A	11/07/00	JP 6309193 A KR 125605 B	04/11/94 01/07/98
US 5748961 A	05/05/98	NONE	
EP 0929034 A2	14/07/99	JP 11203144 A	30/07/99
US 5966539 A	12/10/99	US 5999737 A	07/12/99